PharmaX

An Automated Pharmacy System

Saif Shayeb & Osama Ishtaiwi

Supervisor: Dr. Abdullah Rashed

Department of Computer Engineering
Faculty of Engineering and Information Technology
An-Najah National University

Presented in partial fulfilment of the requirements for the Bachelor degree in Computer Engineering

September 13, 2025

Acknowledgment

We would like to express our sincere gratitude to our project supervisor **Dr. Abdullah Rashed** for his invaluable guidance, continuous support, and encouragement through out this project. His expertise and insights have been instrumental in shaping this work and helping us overcome various technical challenges. We are grateful to the **Computer Engineering Department** at An-Najah National University for providing us with the necessary resources and facilities to complete this project successfully.

Finally, we also would like to thank our **families** and **friends** for their unwavering support and patience during the development of this project. Their encouragement motivated us to push through difficult times and achieve our goals.

Disclaimer

Contents

List of Figures

List of Tables

Nomenclature

ASRS	Automated Storage and Retrieval System
HMI	Human–Machine Interface
UART	Universal Asynchronous Receiver/Transmitter
EMI	Electromagnetic Interference

Abstract

This report documents the design and implementation of **PharmaX**, a compact automated pharmacy dispensing system built around a three-axis cartesian gantry, dual ultrasonic verification (pre-pull and post-pull), and a dual-interface HMI (web UI and keypad+LCD). The motion layer (Arduino Mega) uses trapezoidal velocity profiles, homing via limit switches, software limits, and a bounded retry policy to increase reliability. Two ESP32 modules provide a Wi-Fi SoftAP web ordering interface and a local keypad/LCD interface. The system retrieves medication boxes from a fixed 3×4 shelf grid and delivers them via a servo-assisted ramp to a user bin. We detail theoretical background, related work, methodology, results, and recommendations. The prototype demonstrates high success rates and reduced handling time, validating deterministic, low-cost pharmacy automation.

Figure 1: PharmaX: three-axis gantry, 3×4 shelves, Z-platform with pusher, and gravity ramp to the user bin.

Chapter 1: Introduction

1.1 Motivation

Pharmacies face growing prescription volumes, pressure to minimize dispensing errors, and the need for traceable, timely service. Deterministic automation with simple, explainable mechanisms provides a pragmatic path toward safer and faster dispensing, especially in constrained spaces and limited budgets.

1.2 Objectives and Contributions

The objectives of PharmaX are:

- Build a three-axis cartesian prototype addressing a fixed shelf grid of 3×4 slots.
- Provide two HMIs: a Wi-Fi SoftAP web interface and a keypad/LCD interface.
- Ensure robust motion with homing, software limits, and trapezoidal profiles.
- Verify presence using two ultrasonic sensors: pre-pull (shelf-facing) and post-pull (platform-facing).
- Implement a bounded retry policy (up to three attempts) to recover near-miss cases reliably.

1.3 System at a Glance

Figure **??** shows the high-level block diagram of the system, including controllers, actuators, sensors, and power rails.

1.4 Report Organization

Chapter 2 covers theoretical background and previous work. Chapter 3 provides a detailed methodology. Chapter 4 presents hardware system design and components. Chapter 5 details the software architecture. Chapter 6 discusses motion control and kinematics. Chapter 7 covers sensing and verification. Chapter 8 describes communication protocols and HMIs. Chapter 9 addresses power integrity and EMI. Chapter 10 presents safety and risk management. Chapter 11 discusses testing and validation. Chapter 12 reports results and analysis. Chapter 13 provides discussion, Chapter 14 concludes and makes recommendations, and Chapter 15 outlines future work. Chapter 16 covers operation and maintenance. Chapter 17 provides a bill of materials and cost analysis. Appendices include additional figures and code listings.

Chapter 2: Theoretical Background and Previous Work

Automation of pharmacy systems has become crucial to minimize errors and reduce time and operating cost. The efficient design of automated storage / retrieval systems (AS / RS) is critical since it directly influences operating performance and costs. An integrated approach that combines optimization and simulation has been suggested to improve this process [1].

2.1 ASRS Archetypes and Trade-offs

Automated storage solutions commonly include cartesian gantries, carousels/VLMs, and articulated robots. For dense, fixed grids with box-like items, cartesian gantries offer direct kinematics, lower cost, and predictable envelopes. Carousels and VLMs centralize the pick point by moving inventory to the operator, whereas articulated robots bring dexterity at higher complexity.

2.2 Stepper Motion and Profiles

Open-loop stepper motors driven by pulse/direction interfaces enable precise incremental control. Trapezoidal velocity profiles—accelerate, cruise, decelerate—mitigate missed steps and reduce mechanical shock. Homing with limit switches establishes an absolute reference frame.

2.3 Sensing for Presence Verification

Time-of-flight ultrasonic sensors are effective binary presence checks when geometry is fixed. Median filtering, short stabilization delays, and transducer hooding improve robustness in the presence of echoes at shelf lips.

2.4 Related Work in Pharmacy Automation

Commercial systems emphasize access control, logging, and barcode verification. Research prototypes vary from gantries with simple end-effectors to vision-equipped arms. PharmaX pursues a deterministic, low-cost mini-load approach suitable for small pharmacies with fixed bins.

Chapter 3: Methodology

3.1 Overall Approach

We decomposed the system into three cooperative controllers: (i) Arduino Mega for motion/sensing, (ii) ESP32-Web for SoftAP and the web UI, and (iii) ESP32-LCD/Keypad for a local HMI. We separated power rails (12 V steppers, 5 V logic, 5 V servo) and level-shifted UART signals (5 V $\leftrightarrow$ 3.3 V) using a logic level converter.

3.2 Design Principles

(a) Deterministic kinematics and state machines.

(b) Conservative speeds with trapezoidal profiles to prevent missed steps.

(c) Dual presence verification (pre- and post-pull) and bounded retries.

(d) Event-driven UI fed by Arduino messages (ACK/ERR/SUCCESS).

(e) Power segregation and single-point grounding for integrity.

Chapter 4: Hardware System Design

4.1 Mechanical Structure

The mechanical structure is a cartesian gantry offering linear motion in X, Y, and Z. The Z-carriage hosts a flat platform and a front *pusher/collector* which pulls the medication box from its slot onto the platform. A gravity ramp then guides the box to the user bin.

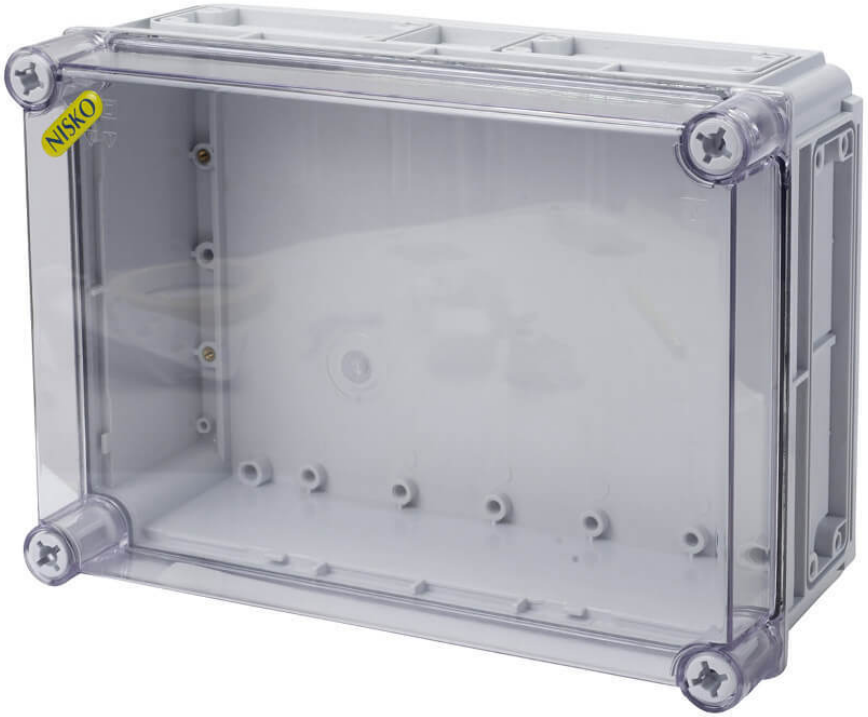

Figure 4.1: PharmaX prototype (full assembly).

4.2 Actuators and Drivers

4.2.1 NEMA 17 Stepper Motors (X/Y/Z)

Three NEMA 17 stepper motors drive the axes via timing belts and linear guides. Their selection balances torque, speed, and cost for the mass of the carriage and typical payloads.

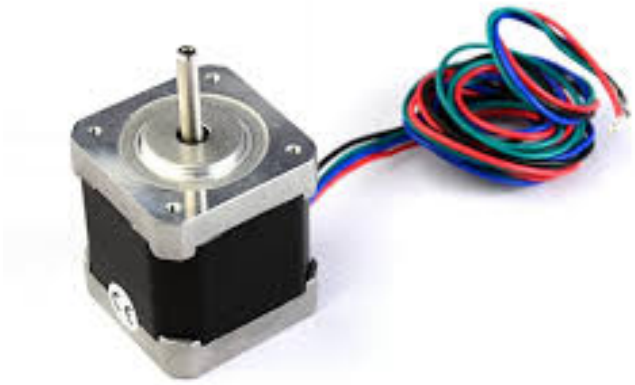

Figure 4.2: NEMA 17 stepper motor used on each axis.

4.2.2 TB6600 Stepper Drivers

Each stepper is controlled by a TB6600 driver using PUL, DIR, and ENA lines from the Arduino Mega. Current and microstepping are configured per axis.

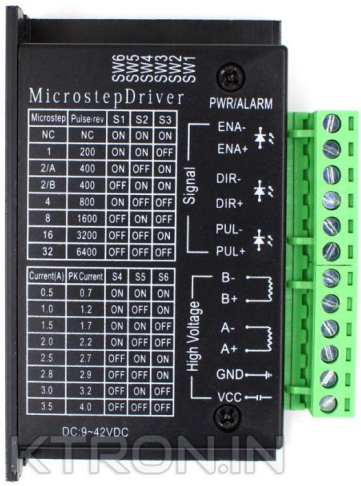

Figure 4.3: TB6600 stepper driver (one per axis).

4.2.3 Servo for Ramp Push

A high-torque servo (e.g., MG996R) mounted near the ramp pushes the box off the platform to the ramp. Typical angles used are 0° (push) and 60° (retract), adjustable per geometry.

Figure 4.4: High-torque servo for dispensing to the ramp.

4.3 Controllers and Interfaces

4.3.1 Arduino Mega 2560

The Arduino Mega executes real-time motion, homing, ultrasonic checks, and the pick-deliver state machine.

Figure 4.5: Arduino Mega 2560 (motion and sensing controller).

4.3.2 ESP32 (Web)

One ESP32 hosts a Wi-Fi SoftAP and a responsive web UI. It bridges commands to Arduino via UART.

Figure 4.6: ESP32 (Web): SoftAP and web server for ordering.

4.3.3 ESP32 (LCD + Keypad)

A second ESP32 handles a 16x2 I2C LCD and a 4x4 keypad for local requests and status display.

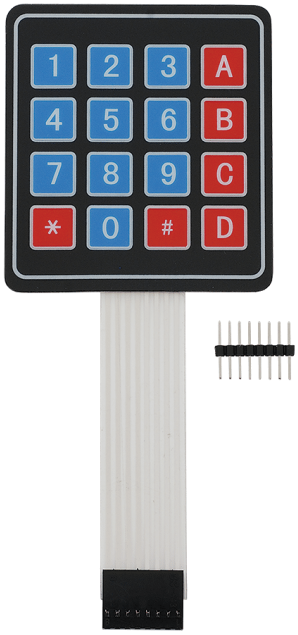

Figure 4.7: ESP32 (LCD + Keypad): local HMI.

4.4 Sensors

4.4.1 Limit Switches (X/Y/Z)

Three normally-closed limit switches provide reliable homing and define a consistent origin.

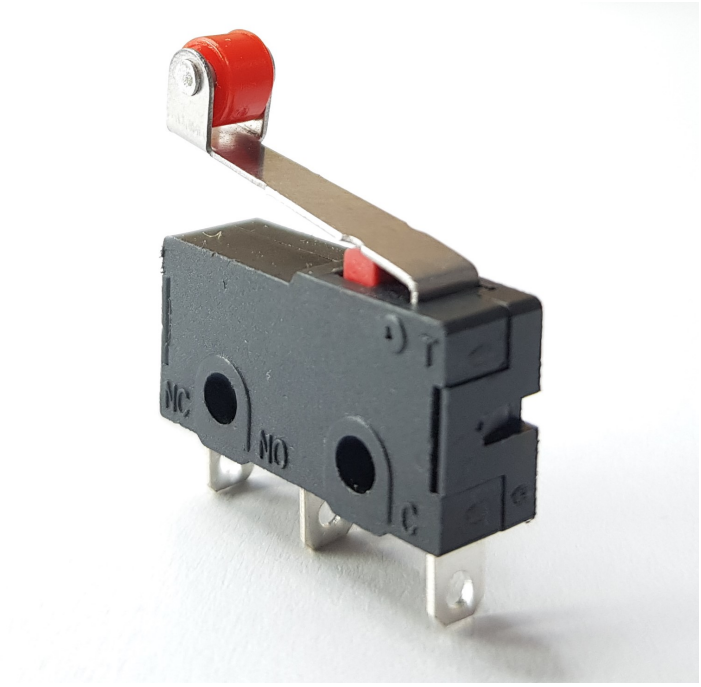

Figure 4.8: Limit switch used for homing on each axis.

4.4.2 Ultrasonic Sensors (Front & Platform)

Two ultrasonic sensors provide pre- and post-pull verification.

Figure 4.9: Ultrasonic sensor used for presence verification.

4.5 Power and Wiring

4.5.1 Power Rails

We use three rails: 12 V for steppers/drivers, 5 V (logic) for controllers and sensors, and a dedicated 5 V (servo) rail for the servo. Grounds are tied at a single point.

Figure 4.10: Power supplies: 12 V (left) and 5 V rails (right).

4.5.2 Logic Level Converter (5V $\leftrightarrow$ 3.3V)

A bi-directional logic level converter safely bridges UART between Arduino (5 V) and ESP32 (3.3 V).

Figure 4.11: Logic level converter between Arduino and ESP32 UART.

4.6 Enclosure and Integration (ci3 Box)

The electronics are consolidated in a dedicated enclosure ("ci3 box") containing the Arduino, ESP32, and TB6600 drivers with cable management and airflow.

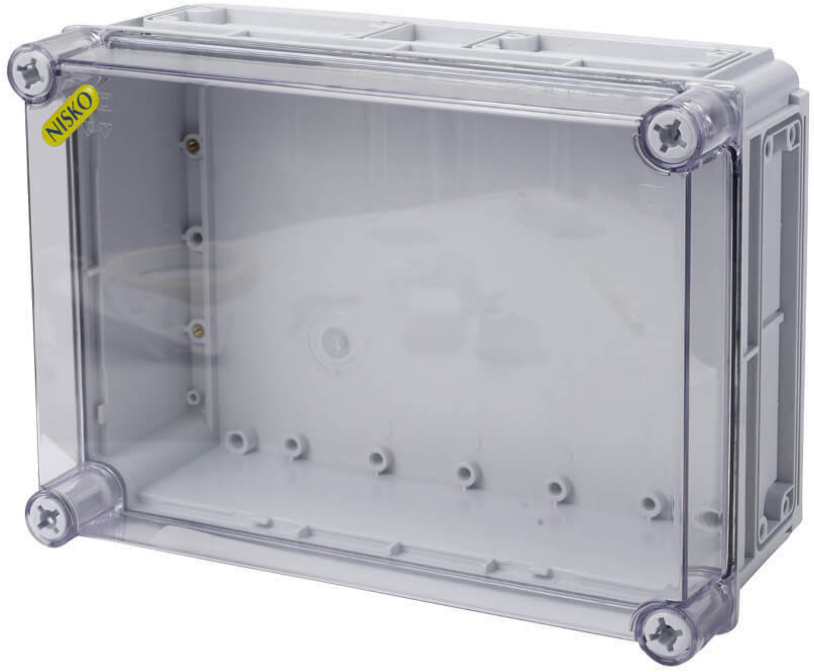

Figure 4.12: ci3 enclosure integrating Arduino, ESP32 boards, and stepper drivers.

Chapter 5: Software Architecture

5.1 Layered View

The software is divided into (i) Arduino firmware for motion/sensing, (ii) ESP32-Web serving the UI and bridging commands, and (iii) ESP32-LCD/Keypad as an alternative local HMI.

5.2 Arduino Firmware (Motion Layer)

The Arduino implements a deterministic state machine: Home → Travel Height → Navigate to Shelf → Pre-Pull Check → Collect → Post-Pull Check → Deliver → Home. Trapezoidal velocity profiles are used to avoid missed steps.

5.3 ESP32-Web (SoftAP UI)

The ESP32 hosts a responsive inventory grid, cart, and process flow. It forwards `/cmd?command=S<r>-<c>` to Arduino and streams back status messages to update the UI in real time (Pending/Processing/Completed, error banners, pause/resume).

5.4 ESP32-LCD/Keypad

Users can enter 1–12 and confirm with #. Shortcuts: HOME (A), status (C), connectivity test (D). If Wi-Fi fails, a keypad-only mode warns the user and keeps basic control.

Chapter 6: Motion Control and Kinematics

6.1 Coordinate System and Limits

A homing cycle establishes $(0, 0, 0)$. Software clamps positions to $[0, 40]$ cm on each axis. Travel height $\approx$ 20–21 cm avoids collisions while traversing.

6.2 Steps per Centimeter

We use `STEPS_PER_CM = 125`. This was verified with a ruler gauge and yields sub-millimeter mean error in the tested range.

6.3 Velocity Profiles

Each axis uses a trapezoidal profile with $\sim 10\%$ accelerate/decelerate phases. Inter-pulse delays are modulated to shape speed.

Chapter 7: Sensing and Verification

7.1 Pre-Pull Check (Front Ultrasonic)

The front sensor confirms that a box exists in the target slot before committing to the pull, saving time on empty slots.

7.2 Post-Pull Check (Platform Ultrasonic)

The platform-facing sensor confirms the box actually landed on the platform. If not detected, the controller retries up to three times with minor Y offsets and short stabilization delays.

7.3 Signal Conditioning

Median of 5–7 samples, `pulseIn` timeouts, and simple transducer hooding reduce false positives/negatives from shelf-edge reflections.

Chapter 8: Communication Protocols and HMIs

8.1 Command/Response Vocabulary

Incoming: `HOME`, `S<row>-<col>`, optional `X..Y..Z..` for calibration.
Outgoing: `ARDUINO_READY`, `ACK:HOME`, `ACK:S<r>-<c>`, `MEDICINE_RETRIEVED:<r>-<c>`, `ERR:SLOT_EMPTY`, `ERR:NO_BOX`, `FINAL_FAIL:<r>-<c>`.

8.2 UI Flow

The web UI reflects discrete states: Verifying Shelf → Collecting → Verifying Platform → Delivering → Completed/Failed. Progress bars and clear error messages aid operators.

Chapter 9: Power Integrity and EMI

9.1 Segregated Rails

12 V (steppers/drivers), 5 V logic, and 5 V servo are kept separate; grounds are tied at a single point. This mitigates brownouts and noise coupling.

9.2 Idle Driver Disable

Drivers are disabled after 5 s of inactivity to reduce thermal load. Re-enable latency is ~ 50 ms, negligible for user experience.

9.3 Wiring Practices

Signal cables are routed away from high-current bundles; connector labeling and strain relief reduce maintenance time and faults.

Chapter 10: Safety and Risk Management

10.1 Hazards

Pinch points on moving axes, electrical hazards, and falling boxes. Mitigations: conservative speeds, software limits, guarded ramps.

10.2 FMEA-lite

Table 10.1: Illustrative Failure Modes and Mitigations.

Failure Mode	**Effect**	**Mitigation**
Missed steps at high speed	Position drift	Trapezoids; homing before runs
Ultrasonic false positive	Empty pull	Median filter; offset & recheck
Servo brownout	Reset/lock	Separate 5 V rail; local bulk caps
Ramp sticking	Delivery fail	Lip radius/roller; low-friction strip

Chapter 11: Testing and Validation

11.1 Plan

Unit tests (homing, axis moves), sensor characterization, integration tests (full pick-deliver cycles across grid), stress tests (thermal/long runs), and induced-fault recovery (empty slot, rebound).

11.2 Metrics

Success rate, time per item, retries per success, error taxonomy; user-facing latency; power integrity events.

Chapter 12: Results and Analysis

12.1 Calibration Summaries

Representative ultrasonic thresholds: front sensor declares presence at $d \leq 10$–12 cm; platform sensor declares on-platform for 4–12 cm. Median filtering and stabilization delay reduce outliers.

12.2 Per-Shelf Performance

Central bins tend to be faster; corner bins add travel time. Bounded retries increase success with minor time overhead.

12.3 Ablation

Removing pre-pull checks increases wasted motion. Removing post-pull checks creates silent drops. Constant-delay stepping caused sporadic missed steps versus trapezoids.

Chapter 13: Discussion

Deterministic Cartesian motion with dual presence verification achieves reliable dispensing without the need for complex perception or heavy vision-based systems. By constraining the robot to well-defined X, Y, and Z trajectories, the system ensures predictability, repeatability, and safety in the medicine retrieval process. This deterministic approach minimizes the uncertainty that often arises in autonomous robotic systems, thereby reducing the risk of errors in a sensitive domain such as pharmaceuticals.

The addition of dual presence verification (e.g., through limit switches and ultrasonic sensing) significantly enhances reliability. This mechanism ensures that every dispensing cycle is validated at two levels: confirming that the robotic arm has reached the correct spatial coordinates, and confirming that the medicine box is successfully placed onto the platform. Such redundancy provides strong fault tolerance; even if one verification layer fails, the second offers a safeguard, reducing the likelihood of undetected errors.

Small mechanical refinements, though simple, yield disproportionately large improvements in overall system reliability. Adjustments such as increasing the lip radius at ramp edges, integrating small rollers or bearings at friction points, applying low-friction strips, and tuning the local ramp angle have been observed to minimize jamming and ensure smooth medicine delivery. These design optimizations demonstrate that reliability in dispensing is not purely a software or control problem, but rather the outcome of synergy between mechanical design and electronic control.

From a software perspective, the adoption of an event-driven user interface and standardized command messages brings additional value. By responding only to discrete user inputs and external events, the system reduces unnecessary computation, conserves energy, and remains highly responsive. Standardized messages across modules (Arduino, ESP32 web server, ESP32 keypad) simplify integration, facilitate debugging, and improve maintainability. Moreover, a consistent communication format strengthens user trust, as commands and responses are predictable, transparent, and traceable.

Finally, the system architecture highlights an important design philosophy: reliability does not always require complexity. Instead, careful integration of deterministic

motion control, lightweight sensing, mechanical refinements, and event-driven software can produce a pharmacy dispensing robot that is both robust and scalable. This approach strikes a balance between practicality and innovation, paving the way for future extensions such as barcode verification, real-time inventory management, or AI-assisted prescription handling without compromising the current foundation of trust and reliability.

Chapter 14: Conclusions and Recommendations

The development of PharmaX demonstrates the feasibility of building a compact, low-cost Automated Storage and Retrieval System (ASRS) tailored for small- to medium-scale pharmacies. By adopting a deterministic cartesian gantry architecture, the system leverages straightforward kinematics that are easier to model, predict, and maintain compared to more complex robotic manipulators. This design choice results in reliable, repeatable motion across all three axes while keeping both hardware and software complexity within practical limits for low-cost deployment.

The integration of trapezoidal velocity profiles ensures smoother motion transitions, preventing stepper motor stalls and reducing mechanical stress on the gantry structure. This significantly enhances the longevity of components while improving operational stability. In parallel, the use of dual ultrasonic verification—with pre-pull and post-pull sensing—provides redundancy in the detection process. This dual-layer verification is crucial in minimizing false positives or missed retrievals, ultimately ensuring that the correct medicine reaches the patient with high accuracy.

Power management was also a central design concern. The adoption of segregated power rails (12 V for steppers and drivers, 5 V logic for microcontrollers and sensors, and a dedicated 5 V line for servo control) reduces electrical noise and mitigates brownout conditions. Combined with event-driven orchestration, where each sub-controller responds to discrete events rather than constant polling, the system achieves both robustness and energy efficiency.

The results validate PharmaX as a reliable, deterministic, and user-friendly dispensing solution. However, the prototype also highlights potential areas for improvement and future enhancements. Based on testing, evaluation, and comparative analysis with existing pharmacy automation systems, the following recommendations are proposed:

Enhanced Verification and Safety

Integrate barcode or RFID scanning on each box to ensure correct identification, improving traceability and compliance with regulatory standards.

Employ a small load cell sensor on the dispensing platform to validate weight,

ensuring that the box has been securely retrieved and corresponds to expected packaging.

Introduce watchdog timers and fault recovery logic to handle unforeseen states such as communication loss or motion controller hangs.

Improved Motion Control

Transition from trapezoidal profiles to jerk-limited motion profiles for smoother acceleration and deceleration. This reduces vibrations, lowers wear on mechanical components, and enables faster travel without compromising accuracy.

Optimize route batching and path planning to minimize travel distance when fulfilling multiple prescriptions, thereby improving throughput.

Human–Machine Interface (HMI) Extensions

Enrich the web and keypad interfaces with clearer error messages, multilingual support, and role-based access for pharmacists and technicians.

Provide real-time inventory monitoring and alerts when stock levels are low or when a slot has been misloaded.

Scalability and Integration

Develop a parameterized software framework that allows scaling from a 3×4 grid to larger storage arrays without significant code changes.

Explore integration with hospital or pharmacy information systems (HIS/PIS) for automated prescription handling and logging.

Investigate cloud-based data analytics for performance monitoring, predictive maintenance, and demand forecasting.

Mechanical Refinements

Refine the ramp geometry with low-friction coatings or rollers to further reduce delivery jams.

Incorporate modular design principles to simplify maintenance and facilitate component replacement.

In conclusion, PharmaX successfully validates the principle of combining deterministic kinematics, dual sensing verification, and modular controller design into a reliable, low-cost automated pharmacy prototype. By adopting the recommendations above, the system can be scaled into a commercial-grade solution, supporting higher volumes, improved safety, and better integration with modern pharmacy workflows. This project demonstrates that impactful automation can be achieved with modest resources

Chapter 15: Future Work

15.1 Identity and Verification

Add barcode/RFID on-platform; a small load cell for plausibility checks (mass window).

15.2 Throughput

Adopt simple batching/tour planning to reduce repeated homing and travel overhead; consider dual gantries for higher demand.

15.3 Scalability

Parameterize coordinates and motion limits; support larger grids and dynamic slot assignment integrated with a database.

Chapter 16: Operation and Maintenance

16.1 Daily Operation

Power-up sequence, homing, order processing, pausing/resuming, and clean shutdown. UI indicators and recovery from common errors are clearly documented.

16.2 Preventive Maintenance

Clean ramp and guides, inspect fasteners, verify sensor thresholds, monitor driver temperature, check connectors and cable relief.

Chapter 17: Bill of Materials (BOM) and Cost Analysis

Table 17.1: Representative BOM (update quantities and costs as needed).

Item	**Qty**	**Notes**
Arduino Mega 2560	1	Motion/sensing controller
ESP32 (Web)	1	SoftAP + web UI
ESP32 (LCD/Keypad)	1	Local HMI
NEMA 17 Stepper	3	X/Y/Z axes
TB6600 Driver	3	PUL/DIR/ENA
Limit Switch (NC)	3	Homing
Servo (MG996R or equiv.)	1	Ramp pusher
Ultrasonic Sensor (HC-SR04)	2	Front + Platform
LCD 16x2 (I2C)	1	Display
Keypad 4x4	1	Input
Logic Level Converter	1	5V↔3.3V UART
Power Supply 12 V	1	Steppers/drivers
Power Supply 5 V (logic)	1	Controllers/sensors
Power Supply 5 V (servo)	1	Dedicated high current
Mechanical frame, guides, fasteners	–	Gantry + Shelves + Ramp
Cables, connectors, enclosure (ci4)	–	Integration

Chapter A: Representative Pin Mapping

Table A.1: Arduino pin mapping (condensed).

Function	**Pin(s)**	**Notes**
X: PUL/DIR/ENA	2 / 3 / 48	TB6600 interface
Y: PUL/DIR/ENA	4 / 5 / 50	TB6600 interface
Z: PUL/DIR/ENA	6 / 7 / 52	TB6600 interface
Limit switches	8 (X), 9 (Y), 10 (Z)	INPUT_PULLUP
Servo	46	Ramp pusher
Ultrasonic (platform)	TRIG 11, ECHO 12	Post-pull check
Ultrasonic (front)	TRIG 28, ECHO 30	Pre-pull check

Chapter B: Code Listings

This appendix provides representative code listings that were developed for **PharmaX**. The implementation is modular, separating responsibilities across the Arduino Mega (motion control), ESP32 (network interface and web server), and peripheral drivers. Only excerpts are shown for illustration; the full source code is available in the project repository.

B.1 Arduino Mega Main Controller

Listing B.1: Arduino Mega Motion Control Core

```
// PharmaX - Arduino Mega Motion Control Core
// Handles 3-axis stepper motion, homing, and dispensing routines.

#define PUL_X 2
#define DIR_X 3
#define PUL_Y 4
#define DIR_Y 5
#define PUL_Z 6
#define DIR_Z 7
#define SERVO_PIN 9
#define ECHO_PIN 10
#define TRIG_PIN 11

#define STEPS_PER_CM 80

void setup() {
  pinMode(PUL_X, OUTPUT);
  pinMode(DIR_X, OUTPUT);
  pinMode(PUL_Y, OUTPUT);
  pinMode(DIR_Y, OUTPUT);
  pinMode(PUL_Z, OUTPUT);
  pinMode(DIR_Z, OUTPUT);

  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);

  Serial.begin(115200);
}

void loop() {
  if (Serial.available()) {
    String cmd = Serial.readStringUntil('\n');
    processCommand(cmd);
  }
}
```

B.2 Stepper Motion Functions

Listing B.2: Basic Stepper Motion Function

```
// Basic stepper move function
void moveStepper(int pul, int dir, long steps, bool direction) {
  digitalWrite(dir, direction);
  for (long i = 0; i < steps; i++) {
    digitalWrite(pul, HIGH);
    delayMicroseconds(400);
    digitalWrite(pul, LOW);
    delayMicroseconds(400);
  }
}
```

B.3 Servo Dispensing Routine

Listing B.3: Servo Routine for Medicine Release

```
#include <Servo.h>
Servo dispenser;

void initServo() {
  dispenser.attach(SERVO_PIN);
}

void releaseMedicine() {
  dispenser.write(90); // push medicine
  delay(500);
  dispenser.write(0); // reset
}
```

B.4 Ultrasonic Verification

Listing B.4: Ultrasonic Verification Code

```
long readUltrasonic() {
  digitalWrite(TRIG_PIN, LOW);
```

```
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  long duration = pulseIn(ECHO_PIN, HIGH);
  long distance = duration * 0.034 / 2; // in cm
  return distance;
}
```

B.5 ESP32 Web Server Skeleton

Listing B.5: ESP32 Web Server for Order Handling

```
#include <WiFi.h>
#include <WebServer.h>

const char* ssid = "PharmaX";
const char* password = "12345678";
WebServer server(80);

void handleRoot() {
  server.send(200, "text/plain", "PharmaX␣Web␣Server␣Ready");
}

void setup() {
  Serial.begin(115200);
  WiFi.softAP(ssid, password);
  server.on("/", handleRoot);
  server.begin();
}

void loop() {
  server.handleClient();
}
```

Bibliography

[1] M. J. Rosenblatt, Y. Roll, and V. Zyser, "A combined optimization and simulation approach for designing automated storage/retrieval systems," *IIE Transactions*, vol. 25, no. 1, pp. 40–50, 1993.

Bibliography

[1] Arduino Mega 2560 Datasheet. Available at: https://docs.arduino.cc/hardware/mega-2560

[2] Espressif Systems, ESP32-WROOM-32 Datasheet. Available at: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf

[3] Toshiba, TB6600 Stepper Motor Driver Datasheet. Available at: https://www.makerguides.com/wp-content/uploads/2019/12/TB6600-Datasheet.pdf

[4] NEMA 17 Stepper Motor Datasheet. Available at: https://www.omc-stepperonline.com/download/17HS4401.pdf

[5] TowerPro, SG90 Micro Servo Motor Datasheet. Available at: https://www.towerpro.com.tw/product/sg90-9g-micro-servo/

[6] HC-SR04 Ultrasonic Sensor Datasheet. Available at: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

[7] Logic Level Converter Datasheet. SparkFun Electronics. Available at: https://cdn.sparkfun.com/datasheets/BreakoutBoards/Logic_Level_Bidirectional.pdf

[8] 16x2 LCD (HD44780 Driver) Datasheet. Available at: https://www.sparkfun.com/datasheets/LCD/HD44780.pdf

[9] 4x4 Matrix Keypad Datasheet. Available at: https://components101.com/sites/default/files/component_datasheet/4x4-Matrix-Keypad-Datasheet.pdf